Smart Battery System Specifications Addendum

Revision 0.95
May 14, 1996

Smart Battery System Specifications Addendum

Please forward questions and comments regarding this specification to:

Smart Battery System Specifications Addendum
Intel MHPG MPE
SC9 - 11
Santa Clara, CA 95052

Phone	**(408) 765 - 6288**
FAX	**(408) 765 - 6553**
Email	**matt_fruin@ccm.sc.intel.com**

Intel thanks Duracell Corporation, Maxim Integrated Products, Benchmarq Electronics, National Semiconductor, Linear Technology, and Microchip Technology for their contributions to this document.

TABLE OF CONTENTS

Revision History

Revision Number	Date	Author	Notes
0.8	4/5/96	M. Fruin	initial public release
0.9	4/12/96	M. Fruin	second public release
0.91	4/19/96	M. Fruin	intermediate release before 0.92
0.92	4/26/96	M. Fruin	third major public release, including changes from third working group meeting
0.95	5/13/96	M. Fruin	fourth major public release, includes all discussions from all working group meetings
1.0	5/24 (planned)	M. Fruin	to be ratified / amended at May PlugFest

1. Introduction

The System Management Bus (SMBus) is a two-wire interface through which simple power-related chips can communicate with the rest of the system; it uses I2C as its backbone. Intel and Duracell along with industry partners have developed specifications for the operation of the bus itself as well as some of the components which communicate over that bus. As devices implementing the various specifications have been developed, it has become apparent that a number of ambiguities exist in the current specifications of SMBus and components utilizing SMBus. This document attempts to address these issues and close any remaining holes in the specifications.

1.1. Background and Scope

Many of the vendors developing SMBus devices met in January 1996 for a two-day “plugfest” where interoperability of components was tested. Out of this conference a number of “working groups” were formed to address open issues in the various specifications. Since that time the communications and operations group has met several times and come to a consensus on some of the details of the bus and device specifications.

This document supplements the existing specifications; it is not intended to stand on its own. It is understood, however, that the descriptions here supersede those in the specifications in cases of conflict, and these descriptions will be incorporated into the next revision of the specifications. This document will be revised as new issues are raised and settled.

1.2. Affected Specifications

The specifications this document specifically address include the following:

- System Management Bus Specification, Revision 1.0, February 15, 1995
- Smart Battery Data Specification, Revision 1.0, February 15, 1995
- Smart Battery Charger Specification, Revision 0.95a, February 15, 1995
- Smart Battery Selector Specification, Revision 0.9, 4/13/95

2. Smart Battery Charger Specification

This section addresses issues related to the Smart Battery Charger Specification, Revision 0.95a, dated February 15, 1995. These guidelines are suggested for V0.95a-compliant Smart Battery Chargers. They will be incorporated into the V1.0 Smart Battery Charger specification and will become requirements except as noted.

2.1. New Smart Battery Charger Registers

To support implementations of a charger and selector in the same component, an extension to the charger address space has been defined. The new registers are optional and only needed for charger components which also include selector functionality. If implemented, the charger's SelectorState() and SelectorPresets() functions match the corresponding functions described in the Smart Battery Selector Specification (and amended below). The mapping of the Smart Battery Selector functions into the Charger address space may be accomplished by OR'ing the desired Selector command code with 0x20 and then sending it to the Smart Battery Charger's SMBus address. Therefore, for a charger / selector component, the function code for SelectorState() is 0x21, and for SelectorPresets(), 0x22. Note that the charger's defined optional manufacturer's extended commands are located in the range of 0x3c-3f; it is expected that this area would be shared by both the charger and selector in a dual-function component.

2.2. Charger Brown-Out Conditions

It is possible there will arise situations where the system attempts to charge a battery while the system is also being powered from the AC power supply. In these cases, the charger shall not draw so much power that the system's power source is compromised. The charger can, at its option, choose to charge the battery at a lower rate automatically or abort charging entirely.

2.3. Out-Of-Regulation Status Bits

Two new optional bits have been added to the charger's ChargerStatus() register: bit 2 = VOLTAGE_NOTREG, where a 1 indicates the charger detects that the requested voltage in the ChargingVoltage() register is not in regulation, and bit 3 = CURRENT_NOTREG, where a 1 indicates the charger detects that the requested current in the ChargingCurrent() register is not in regulation. These bits are not defined when the charger is disabled. These bits are optional.

2.4. "Lost" Communications Time-out

A new safety feature has been added to the Smart Charger specification requiring a charger to stop charging a battery when the charger has "lost" its communications link to the battery being charged. The charger determines this by not receiving a new ChargingCurrent() or ChargingVoltage() command within a time-out period (Minimum 140 seconds, Maximum 210 seconds). The charger can resume charging once it has received both a new ChargingCurrent() and a new ChargingVoltage() command.

2.5. Charger Leakage Current

Text will be added to the Smart Battery Charger Specification describing why leakage current into a battery and out of a battery should be minimized; however, min and max values for these parameters will not be specified.

2.6. Start-Up Operation Clarification

Upon start-up or at any other power-on condition (eg, after POR_RESET has been set), the charger may supply "wake-up" charge to the battery at the charger's maximum voltage and up to 100 mA, if the following conditions are satisfied: AC power is present, a battery is present, the thermistor value is determined to be greater than 3k ohms or less than 500 ohms, the INHIBIT_CHARGE bit is not set, and the time-out period has not expired (Minimum 140

seconds, Maximum 210 seconds). Once the time-out period expires, the charger can optionally continue to "wake-up" charge if it measures the battery thermistor greater than 3k ohms and less than 30k ohms. See section 2.8 for a more complete description of all charger operational modes.

It is expected that at least some constant-voltage battery packs will use a resistor < 500 ohms to indicate that these batteries do not want extended "wake-up" charging.

This functionality covers the following expected system scenarios:

- *Constant current (eg, NiMH) smart battery, intelligence "awake"* **--** battery, if full, sends AlarmWarning() message to abort charging, or sends ChargingCurrent() and ChargingVoltage() commands asking to be charged at a higher rate.
- *Constant current smart battery, intelligence "asleep"* **--** it could take up to 15 minutes of "wake-up" charging to raise the battery pack's voltage enough for the intelligence to wake up. The charger can wait until that occurs, or the host could conclude it is charging a "dumb" battery and begin charging at a higher rate.
- *Constant current "dumb" battery* **--** wake-up charging is acceptable, or the host could decide to charge the battery at a higher rate.
- *Constant voltage (eg, Li-ion) smart battery, intelligence "awake"* **--** battery sends an AlarmWarning() message asking the charger to stop charging, or sends ChargingCurrent() and ChargingVoltage() commands requesting constant voltage charging.
- *Constant voltage smart battery, intelligence "asleep" (also implies that a deep-discharge protection FET is open-circuited as well)* **--** the voltage at the terminals of the battery pack should be enough to wake up the battery's microcontroller, which then sends charging voltage and current messages to the charger and re-enables its protection device.
- *Constant voltage dumb battery* **--** not supported.

Wake-up charging is useful as a mechanism for starting communication with batteries whose voltage is so low that the internal microcontroller does not have enough power to operate. If such a mechanism did not exist, a deeply discharged battery would never generate charging current and voltage commands, and hence would never get charged. By providing voltage at the battery terminals, it is expected that the microcontroller will be restarted and can then communicate with the system, either to stop wake-up charging or to program faster charging.

2.7. Thermistor Ranges

As an additional required safety feature, the charger must not charge a battery when it senses the thermistor between 500 and 3k ohms. A thermistor in a NiMH battery (eg) would enter this range if it got too hot; the thermistor in a Li-ion battery (eg) could be set to this range in an emergency alarm condition. (Note that using the thermistor should be a last line of communication; a battery should first send an appropriate message if it wishes to abort charging.)

The defined thermistor ranges are the following:

Thermistor	Charger Status Bits	Description	wake-up charge	controlled charge	Notes
0 to 500Ω	THERM_UR, THERM_HOT	underrange	recommended for initial timeout period	allowed	Charger can "wake-up" charge for timeout period; controlled charge allowed.
500 to 3kΩ	THERM_HOT	hot	not allowed	not allowed	Fail-safe charge termination -- charger must not supply current
3k to 30kΩ	(none)	normal range	recommended indefinitely	allowed	Charger can "wake-up" charge indefinitely; controlled charge allowed.
30k to 100kΩ	THERM_COLD	cold	allowed for initial timeout period	allowed	Charger can "wake-up" charge for timeout period; controlled charge allowed.
above 100kΩ	THERM_OR, THERM_COLD	overrange	not allowed	not allowed	Charger can default to voltage = 0 to battery pack voltage, current as small as possible (<< 10mA)

2.8. Charger Operational Modes Clarifications

The table below clarifies the charger's operation under all conditions.

#	Condition	Action
1	The charger is in its power-on state AND a battery is present AND AC is present AND INHIBIT_CHARGE = 0 AND THERM_HOT = 0.	The charger can optionally "wake-up" charge the battery at up to 100mA indefinitely.
2	The charger is in its power-on state AND a battery is present AND AC is present AND INHIBIT_CHARGE = 0 AND THERM_HOT = 1 AND THERM_UR = 1.	The charger can optionally "wake-up" charge the battery at up to 100mA until the timeout period expires (140-210 seconds).
3	The charger is applying "wake-up" charge to the battery AND (the battery is removed OR AC power is removed OR a 1 is written to POR_RESET).	The charger is set to its power-on default state.
4	The charger is applying "wake-up" charge to the battery (from condition 1 above) AND THERM_HOT changes from 0 to 1.	The charger stops charging the battery. It cannot "wake-up" charge again until it gets a POR_RESET, AC is removed, or the battery is removed. It can supply "controlled charge" to the battery if THERM_HOT is reset to 0, INHIBIT_CHARGE = 0, and both ChargingCurrent() and ChargingVoltage() commands are received.
5	The charger is applying "wake-up" charge	The charger stops charging the battery. It

	to the battery (from condition 2 above) AND THERM_UR changes from 1 to 0.	cannot “wake-up” charge again until it gets a POR_RESET, AC is removed, or the battery is removed. It can supply “controlled charge” to the battery if THERM_UR is set to 1, INHIBIT_CHARGE = 0, and both ChargingCurrent() and ChargingVoltage() commands are received. (Dangerous for battery to assume this?)
6	The charger is applying “wake-up” charge to the battery AND an AlarmWarning() message is received with any bit in the upper nibble set.	The charger stops charging the battery. It cannot “wake-up” charge again until it gets a POR_RESET, AC is removed, or the battery is removed. It can supply “controlled charge” to the battery when THERM_HOT = 0 or both THERM_HOT and THERM_UR = 1, INHIBIT_CHARGE=0, and both ChargingCurrent() and ChargingVoltage() commands are received.
7	The charger is applying “wake-up” charge to the battery AND INHIBIT_CHARGE is set to 1.	The charger stops charging the battery. The timer continues to run. The charger can resume “wake-up” charging if INHIBIT_CHARGE is reset to 0 and THERM_HOT = 0 or THERM_HOT = 1 and THERM_UR = 1.
8	The charger is applying “wake-up” charge to the battery (or had been but has been stopped), AND a ChargingCurrent() command is received AND a ChargingVoltage() command is received AND INHIBIT_CHARGE = 0 AND (THERM_HOT = 0 OR THERM_HOT = 1 and THERM_UR = 1).	The charger will supply “controlled charge” to the battery as specified in the Current and Voltage commands.
9	The charger is supplying “controlled charge” to the battery AND (an AlarmWarning() message is received with any bit in the upper nibble set OR THERM_UR = 1).	The charger stops charging the battery. It can resume “controlled charge” when THERM_UR = 0, INHIBIT_CHARGE = 0, and both ChargingCurrent() and ChargingVoltage() commands are received.
10	The charger is supplying “controlled charge” to the battery AND no new ChargingCurrent() or ChargingVoltage() command is received for a time-out period (140-210 seconds).	The charger stops charging the battery. It can resume “controlled charge” when THERM_UR = 0, INHIBIT_CHARGE = 0, and both a new ChargingCurrent() and a new ChargingVoltage() command is received.
11	The charger is supplying “controlled charge” to the battery AND THERM_HOT changes from 0 to 1.	The charger stops charging the battery. It can resume “controlled charge” when THERM_UR = 0, THERM_HOT = 0, INHIBIT_CHARGE = 0, and both ChargingCurrent() and ChargingVoltage() commands are received.
12	The charger is supplying “controlled charge” to the battery AND THERM_UR changes from 1 to 0.	The charger stops charging the battery. It can resume “controlled charge” when THERM_UR = 1, THERM_HOT = 1,

		INHIBIT_CHARGE = 0, and both ChargingCurrent() and ChargingVoltage() commands are received.
13	The charger is supplying “controlled charge” to the battery AND INHIBIT_CHARGE is set to 1.	The charger stops charging the battery. It can resume charging when INHIBIT_CHARGE is cleared to 0. The charger will continue to accept new ChargingVoltage() and ChargingCurrent() commands while INHIBIT_CHARGE=1.
14	The charger is supplying “controlled charge” to the battery AND (ChargingCurrent() is set to 0 OR ChargingVoltage() is set to 0 or a 1 is written to RESET_ZERO).	The charger stops charging the battery. It can resume charging the battery when new ChargingCurrent() and ChargingVoltage() messages are received and THERM_UR and INHIBIT_CHARGE remain cleared. It will not revert to “wake-up” charge.
15	The charger is supplying “controlled charge” to the battery AND (the battery is removed OR AC power is removed OR a “1” is written to POR_RESET).	The charger is set to its power-on default state.
16	The charger detects no battery present.	The charger can default to 0 to expected battery pack voltage at 0 to 10 mA.

Writing a 1 to RESET_ZERO or POR_RESET will cause various actions but will not set the bit.

The Smart Battery Charger specification will be changed to include the definition that the INHIBIT_CHARGE bit is intended to be the host’s mechanism for suspending charge. The battery is explicitly not allowed to write to this bit.

2.9. Charger Interrupt Mechanism

The specification will be changed to include an optional interrupt mechanism for a smart charger to indicate to the system a change in its status, for example, battery insertion or removal or AC present. Having such a mechanism is optional but desirable, especially for single-battery systems which do not implement a Smart Battery Selector which would indicate changes in the system’s power status.

2.10. Charging “Dumb” Batteries

It is acceptable that a Smart Battery Charger host be responsible for charging dumb batteries. In other words, it is not required that the charger determine how to charge a battery which does not broadcast ChargingCurrent() and ChargingVoltage() commands. In any event, care must be taken such that any non-battery-initiated charging is safe, and not mistakenly charging a damaged smart battery (for example, a battery whose microcontroller has failed and cannot communicate).

2.11. “Float” Voltage

Float voltage refers to the voltage output of the charger with no battery connected. There may be a desire to design the system to present a voltage on its terminals to minimize contact arcing when a battery is plugged in. When the charger detects no battery present in the system, it may default to an output of 0 to its full-range voltage and up to 10mA current (though much less is desirable).

3. Smart Battery Selector Specification

This section addresses issues related to the Smart Battery Selector Specification, Revision 0.9, dated 4/13/95. They will be incorporated into the next release of the Smart Battery Selector specification and will become requirements except as noted.

3.1. Power-On Defaults / Start-Up Description

It is expected that a Smart Battery Selector power-up in a consistent and safe state, ideally autonomously setting the SMB_X and POWER_BY nibbles to the power source supplying power to the system and possibly setting CHARGE_BY to an appropriate battery.

3.2. Register Changes and Clarifications

The following clarifies the expected bit values for the SelectorState() function of a Smart Battery Selector.

SelectorState()

Description:
This required function either returns the current state of the selector or allows the state to be altered under programmatic control. The information is broken into four nibbles that report:

- Which battery is communicating with the SMBus Host
- Which battery is (or will be) powering the system
- Which battery is connected to the smart charger
- Which battery(s) are present.

The selector will have a mechanism to notify the system when there is a change in its state, doing so when:

- A battery is added or removed
- AC power is connected or disconnected
- The selector autonomously switches to another battery (for example, the previous one ran low)

Purpose:
Used by the system host to determine the current state of the selector and attached batteries. It also may be used to determine the state of the battery system after the selector notifies the system of a change.
SMBus Protocol: Read or Write Word
Input/Output: word -- bit flags in nibbles mapped as follows:

SMB (r/w) Battery connected to System SMBus (comm.)				**POWER_BY (r/w)** Battery connected to System PS (power & comm.)				**CHARGE (r/w)** Battery connected to Charger (power & comm.)				**PRESENT (r)** Battery(s) Present			
D	**C**	**B**	**A**	**D**	**C**	**B**	**A**	**D**	**C**	**B**	**A**	**D**	**C**	**B**	**A**

SMB_X nibble
The read/write SMB_X nibble is used by the SMBus Host either to select which battery to communicate with or, to determine which battery it is communicating with. Normally, this bit selects the same battery as the POWER_BY_X bit. One and only one bit MUST be set in this register. When writing a value to the POWER_BY_X nibble, the SMB_X nibble MUST have the same bit set.

For example, an application that displays the remaining capacity of all batteries, would use these bits to select each battery in turn and get its capacity. The bits are defined as follows:

0x8###	SMB_D	Host SMBus is connected to Battery D
0x4###	SMB_C	Host SMBus is connected to Battery C
0x2###	SMB_B	Host SMBus is connected to Battery B
0x1###	SMB_A	Host SMBus is connected to Battery A
0x0###		Host SMBus is not connected to a Battery or to an undetermined Battery

All other bit combinations are not allowed.
Note: One and only one bit may be set in this nibble AND will cause any previously established communications path with the SMBus Host established by the POWER_BY_X nibble to be replaced.

POWER_BY_X nibble
The read/write POWER_BY_X nibble is used by the SMBus Host either to select which battery will power the system or to determine which battery is powering the system. It is important to note that any change in the SMB_X nibble is persistent and must be checked to determine that the system is actually communicating with the battery that is supplying power. When selecting which battery will power the system, the POWER_BY_X nibble, the SMB_X nibble MUST have the same bit set.

The bits are defined as follows:

0x#8##	POWER_BY_D	Set system to communicate with and be powered by Battery D
0x#4##	POWER_BY_C	Set system to communicate with and be powered by Battery C
0x#2##	POWER_BY_B	Set system to communicate with and be powered by Battery B
0x#1##	POWER_BY_A	Set system to communicate with and be powered by Battery A
0x#0##	POWER_BY_AC	Set system to be powered by AC power

Note: At most, one bit may be set in this nibble.

CHARGE_X nibble
The read/write CHARGE_X nibble is used by the SMBus Host either to select which, if any, battery will be connected to the charger or to determine which, if any, battery is already connected to the charger. In addition, these bits will be inverted (for example, 0x1 **-** no AC -> 0xE **-**AC present) whenever AC is present. Whether or not AC is present, up to only one bit may be set to indicate which, if any, battery to charge.

The bits are defined as follows:

0x##8#	CHARGE_D	Set to charge Battery D
0x##4#	CHARGE_C	Set to charge Battery C
0x##2#	CHARGE_B	Set to charge Battery B
0x##1#	CHARGE_A	Set to charge Battery A
0x##0#		No battery or undetermined battery connected to charger

Note: Zero or one bits of the CHARGE_X nibble may be set. This nibble will be inverted if AC power is present when the SelectorState() register is read. Writes to this nibble should set at most one bit high, whether or not AC power is present.

PRESENT_X nibble
The read only PRESENT_X nibble is used by the SMBus Host to determine how many and which batteries are present.

The bits are defined as follows:
0x###8 PRESENT_D Battery D is present
0x###4 PRESENT_C Battery C is present
0x###2 PRESENT_B Battery B is present
0x###1 PRESENT_A Battery A is present
Note: Each bit in the PRESENT_X nibble will be set independently to indicate the presence of a battery.

When the power source supplying power to the system (and selected by the POWER_BY nibble) is removed or otherwise becomes unavailable, the Selector should change to another valid source fast enough such that the system's power integrity is not compromised, and update its POWER_BY nibble accordingly (and inform the system that a change has occurred in the system's power subsystem, either through an interrupt line or some other mechanism). However, the SMB_X nibble should not be changed. (It is expected that the host, once it has received and decoded the interrupt from the Selector after the power change has occurred, will correctly set the SMB_X nibble such that the communications and power path are consistent.)

3.3. Mechanism to Mask Registers

Since there are four nibbles in the SelectorState() function, one of which is status, another is status and control, and two are control, it is difficult to ensure a host writing to the register would never overwrite a valid nibble value with outdated information. Therefore, when writing to the SelectorState() function, the host should mask registers by writing all 1's (0xF) to each nibble it doesn't wish to modify. For example, to set the SMB_X nibble to select the first battery but not change any other values, the host would write 0x1FFF to this function.

4. Smart Battery Data Specification

This section addresses issues related to the Smart Battery Data Specification, Revision 1.0, dated February 15, 1995.

4.1. Battery ChargingCurrent() and ChargingVoltage() Frequency

The Smart Battery Data Specification indicates that a battery should broadcast charging current and voltage commands at least every five minutes. This is an error in the specification, and will be changed to at least every minute.

5. System Management Bus Specification

This section addresses issues related to the System Management Bus Specification, Revision 1.0, dated February 15, 1995.

5.1. SMBus Time-out Clarifications

1. The specification of the SMBus timeout parameter, $T_{TIMEOUT}$, is not entirely complete as written. This parameter will be changed to the following: A slave device wishing to cause a timeout should hold the clock line low for at least 26 ms but no more than 35 ms; if the slave does not want to cause a timeout, it must not hold the clock line low for more than 24 ms. A master device must recognize a timeout on the bus when the clock line is held low for 25 ms or more. (NOTE: THESE NUMBERS ARE PRELIMINARY PENDING APPROVAL FROM EXISTING COMPONENT MANUFACTURERS.)

2. The specification of a slave device's cumulative clock low extend time, $T_{LOW:SEXT}$, is a requirement for a slave device to be System Management Bus-compliant, but it is not required nor suggested that a master device track this parameter. Instead, the master need only be track $T_{TIMEOUT}$.

3. There exists a situation where it is possible for a slave device to violate I2C specifications after a timeout has occurred on the SMBus. This case arises when the slave device is transmitting SMBDATA = 0 and detects a timeout caused by the master holding the SMBCLOCK line low for more than 26 ms. The slave should then release both the clock and data lines, but according to a strict interpretation of I2C (upon which SMBus is based), should not release the data line because that would violate I2C timing parameters. Ideally, the master would drive an extra clock cycle prior to generating a stop condition so that the slave could gracefully release the SMBDATA line, but since there are implementations that do not do this it is acceptable for the slave device to release both clock and data lines at the same time after causing the timeout. It is acceptable for a master to generate an extra clock pulse when this condition occurs to allow a slave to release the data line, but not required.

4. It will be emphasized in the specification that whenever possible, a slave device should signal an error by driving a not-acknowledge (NAK) on the bus; generating a timeout should be a last resort since it monopolizes the bus and prevents other devices from using it.

5. A consequence of #3 above, it is recommended that a master that gets timed out should generate a start-stop sequence to reset all the devices on the bus. In addition, all devices should be able to recognize start and stop conditions on the bus, even if such conditions potentially occur out of place.

6. It is possible to implement an SMBus device which does not need to track bus timeouts if it meets the following criteria: (a) the device will not perform any clock stretching of its own, and (b) the device will be reset by a start condition.

5.2. Slave Response Clarification

It is acceptable that a slave device can be "busy" and not complete a transaction (for example, if it is calculating a value). In any case, the slave should always acknowledge its address; it can not acknowledge the command code if it doesn't want to communicate at that time. The master device can retry the transaction if it wants to.

1. Safety Features in the SBS Specifications

Safety has always been a primary design goal in the Smart Battery System specifications. The central concept behind the Smart Battery specifications is locating the primary intelligence of the system inside the battery pack itself, thus enabling the system to be much more accurate in its measurement of battery parameters such as remaining capacity and design voltage, and also allowing the charging algorithm and parameters to be tuned to the battery pack's specific chemistry. By relying on the battery pack's intelligence, a properly designed Smart Battery system can safely charge and discharge any expected battery chemistry. This section will review the safety features of the Smart Battery specifications under the following operating modes: "Controlled" Charging, "Wake-up" Charging, and Discharging.

1. "Controlled" Charging

"Controlled" Charging (ie, charging under control of the battery, usually at a rate faster than "wake-up" charging described below) can only be initiated when and if the battery explicitly requests charging from the Smart Battery Charger device in the system and the charger independently determines it is safe to do so, and can be aborted at any time if the battery or charger detects an error condition. The battery requests charging when it sends two separate messages to the charger indicating its desired charging voltage and charging current; the charger will attempt to supply the requested voltage and current if it measures the thermistor terminal of the battery within an acceptable range. Charging continues as long as new current and voltage messages are sent by the battery to the charger and the thermistor value stays within acceptable limits. Charging will be terminated if any of the following conditions occur:

- a message is sent (by the battery or some other device) terminating charge.
- the resistance of the thermistor moves out of the valid range.
- the charger doesn't receive valid current and voltage messages for a timeout period.

Therefore, to begin charging, both the battery and charger must agree it is safe to do so. First, the battery must explicitly request charging. If the battery detects some error condition (battery pack voltage too high or low, temperature out of acceptable ranges, individual cell voltage shorted, etc...) it won't request charging and charging will never begin. Second, the charger will not begin charging if the thermistor value is not in the valid range.

Furthermore, to continue charging, both the battery and charger must be satisfied it is still safe to do so. First, the battery must continuosly send messages to the charger indicating its requested charging current and voltage. If any error condition occurs in the battery, the battery can send a message to the charger to terminate charge. Or, the battery can simply stop sending messages to the charger and charging will stop after the charger detects the battery is no longer sending out messages. (For the same reason, the charger will stop charging if something happens to the communications channel between the battery and charger, since it will no longer be receiving messages from the battery.) Second, the charger will monitor the battery pack thermistor terminal. If the thermistor moves out of its valid range, the charger will immediately terminate charge. This could happen if the temperature of the battery pack got too hot, or the intelligence in the pack could force the thermistor to that range to indicate an error condition and abort charging immediately.

These mechanisms are independent of and in addition to any protection mechanisms in the battery pack itself, for example, fuses or protection FETs controlled by hard-wired safety circuitry.

2. “Wake-up” Charging

   It is possible for a battery pack to be so depleted that its built-in intelligence does not have enough power to operate. Therefore, the Smart Battery System specifications allow a charger to apply some small amount of charge (< 100mA at the battery pack voltage) to a battery when it is plugged in so that the battery can wake up enough to communicate. This “wake-up” charge will not begin if the thermistor is not in the valid range, and will be terminated when the battery sends a message to the charger, or the thermistor moves out of the valid range, or (depending on the battery chemistry, as indicated by the thermistor value) the timeout period of 140 to 210 seconds expires.

3. Discharging

   The Smart Battery System specifications also have a mechanism for the battery to request that the system stop discharging it. If the battery detects that it no longer wishes to be discharged, it can send a high-priority message to the system asking it to stop drawing power from it. (The specifications allow the battery to continue sending this message at up to 5-second intervals.) The system should process the messages and switch to another power source if available, or go into a low-power state if not. However, there is no hardware mechanism to enforce the battery’s desire to stop providing power through its terminals.